A Learning Automata Based Solution to Service Selection in Stochastic Environments*

Anis Yazidi[1], Ole-Christoffer Granmo[1], and B. John Oommen[2,**]

[1] Dept. of ICT, University of Agder, Grimstad, Norway
[2] School of Computer Science, Carleton University, Ottawa, Canada

Abstract. With the abundance of services available in today's world, identifying those of high quality is becoming increasingly difficult. Reputation systems can offer generic recommendations by aggregating user provided opinions about service quality, however, are prone to "ballot stuffing" and "badmouthing". In general, unfair ratings may degrade the trustworthiness of reputation systems, and changes in service quality over time render previous ratings unreliable.

In this paper, we provide a novel solution to the above problems based on Learning Automata (LA), which can learn the optimal action when operating in unknown stochastic environments. Furthermore, they combine rapid and accurate convergence with low computational complexity. In additional to its computational simplicity, unlike most reported approaches, our scheme does not require prior knowledge of the degree of any of the above mentioned problems with reputation systems. Instead, it gradually learns which users provide fair ratings, and which users provide unfair ratings, even when users unintentionally make mistakes.

Comprehensive empirical results show that our LA based scheme efficiently handles any degree of unfair ratings (as long as ratings are binary). Furthermore, if the quality of services and/or the trustworthiness of users change, our scheme is able to robustly track such changes over time. Finally, the scheme is ideal for decentralized processing. Accordingly, we believe that our LA based scheme forms a promising basis for improving the performance of reputation systems in general.

Keywords: Reputation Systems, Learning Automata, Stochastic Optimization.

1 Introduction

With the abundance of services available in today's world, identifying those of high quality is becoming increasingly difficult. "Reputation Systems" (RSs)

* This work was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

** Chancellor's Professor; Fellow : IEEE and Fellow : IAPR. The Author also holds an Adjunct Professorship with the Dept. of ICT, University of Agder, Norway.

N. García-Pedrajas et al. (Eds.): IEA/AIE 2010, Part III, LNAI 6098, pp. 209–218, 2010.

attracted a lot of attention during the last decade in the academia as well as in the industry. RSs have also emerged as an efficient approach to handle trust in online services, and can be used to collect information about the performance of services in the absence of direct experience.

In this paper we intend to study how experiences can be shared between users in a social network, where the medium of collaboration is a RS. The basic premise, of course, is that it is possible for users to expediently obtain knowledge about the nature, quality and drawbacks of specific services by considering the experiences of other users. The above premise is true if the basis of the decision is accurate, up-to-date and fair. In fact, the social network and the system itself might contain misinformed/deceptive users who provide either unfair positive ratings about a subject or service, or who unfairly submit negative ratings. Such "deceptive" agents, who may even submit their inaccurate ratings innocently, have the effect that they mislead a RS that is based on blindly aggregating users' experiences. Furthermore, when the quality of services and the nature of users change over time, the challenge is further aggravated.

Finding ways to counter the detrimental influence of unfair ratings on a RS has been a focal concern of a number of studies [1,2,3]. Dellarocas [1] used elements from collaborative filtering to determine the nearest neighbors of an agent that exhibited similar ratings on commonly-rated subjects. He then applied a cluster filtering approach to filter out the most likely unfairly positive ratings. Sen and Sajja [2] proposed an algorithm to select a service provider to process a task by querying other user agents about their ratings of the available service providers. The main idea motivating their work is to select a subset of agents, who when queried, maximizes the probability that the majority of the queried agents provide correct reputation estimates. However, comprehensive experimental tests show that their approach is prone to the variation of the ratio of deceptive agents. In contrast, in [4], Witby and Jøsang presented a Bayesian approach to filter out dishonest feedback based on an iterated filtering approach. The authors of [5] proposed a probabilistic model to assess peer trustworthiness in P2P networks. Their model, which in one sense is similar to ours, differs from the present work because the authors of [5] assume that a peer can deduce the trustworthiness of other peers by comparing its own performance with reports of other peers about itself. Though such an assumption permits a feedback-evaluating mechanism, it is based on the fact that peers provide services to one another, thus permitting every party the right to play the role of a service provider and the service consumer (a reporting agent). Our approach, which we briefly describe in the next section makes a clear distinction between these parties – the service provider and the reporting

In this paper, we provide a novel solution to the above problems based on Learning Automata, which can learn the optimal action when operating in unknown stochastic environments [6]. Furthermore, they combine rapid and accurate convergence with low computational complexity. In additional to its computational simplicity, unlike most reported approaches, our scheme does not require prior knowledge of the *degree* of any of the above mentioned problems

with RSs. Rather, it adaptively, and in an on-line manner, gradually learns the identity and characteristics of the users who provide fair ratings, and of those which provide unfair ratings, even when these are a consequence of them making unintentional mistakes.

Unlike most existing reported approaches that only consider the feedback from "fair" agents as being informative, and which simultaneously discard the feedback from "unfair" agents, in our work we attempt to intelligently combine (or fuse) the feedback from fair and deceptive agents when evaluating the performance of a service. Moreover, we do not impose the constraint that we need *a priori* knowledge about the ratio of deceptive agents. Consequently, unlike most of existing work, that suffer from a decline in the performance when the ratio of deceptive agents increases, our scheme is robust to the variation of this ratio. This characteristic phenomenon of our scheme is unique.

2 Modeling the Problem

Consider a population of L services (or service providers), $\mathcal{S} = \{S_1, S_2, \ldots, S_L\}$. We also assume that the social network (or pool of users) consists of N parties (synonymously called "agents") $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$. Each service S_l has an associated quality, which, in our work is represented by an "innate" probability of the service provider performing exceptionally well whenever its service is requested by an agent. This probability is specified by the quantity θ_l, assumed to be unknown to the users/agents. For a given interaction instance between user agent u_i and service S_l, let x_{il} denote the performance value, which, for the sake of formalism, is assumed to be generated from a distribution referred to as the *Performance Distribution* of S_l. After the service has been provided, the user/agent u_i observes the performance x_{il}, where $x_{il} \in \{0, 1\}$. We assume that '0' denotes the lowest performance of the service, while '1' denotes its highest performance.

At this juncture, after the agent has experienced the quality of the service, he communicates his experience to the rest of the network. Let y_{il} be the report that he transmits to other agents after he experiences x_{il}. Obviously[1], $y_{il} \in \{0, 1\}$. To do this, we assume that agent u_i communicates his experience, x_{il}, truthfully to other agents in the population, with a probability p_i. In other words, p_i denotes the probability that agent u_i is not misreporting his experience. For ease of notation, we let $q_i = 1 - p_i$, which represents the probability that agent u_i does indeed misreport his experience. Clearly, $p_i = Prob(x_{il} = y_{il})$.

Observe that as a result of this communication model, a "deceptive" agent will probabilistically tend to report low performance experience values for high performance services and *vice versa*. Our aim is to partition the agents as being true/fair or deceptive.

Formally, the Agent-Type Partitioning Problem ($ATPP$), can be stated as follows: A social network consists of N agents, $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, where each

[1] We mention, in passing, that other researchers, have used the notation y_{il} to signify the rating of the service.

agent u_i is characterized by a fixed but unknown probability p_i of him reporting his experience truthfully. The ATPP involves partitioning $\mathcal{U}$ into 2 (mutually exclusive and exhaustive) groups so as to obtain a 2-partition $\mathbb{G}_k = \{G_i \ | i = 1, 2\}$, such that each group, G_i, of size, N_i, exclusively contains only the agents of its own type, i.e., which either communicate truthfully or deceptively.

To simplify the problem, we assume that every p_i can assume one of two possible values from the set $\{p_d, p_f\}$, where $p_d < 0.5$ and $p_f > 0.5$. Then, agent u_i is said to be fair if $p_i = p_f$, and is said be deceptive if $p_i = p_d$.

Based on the above, the set of fair agents is $\mathcal{U}_f = \{u_i | p_i = p_f\}$, and the set of deceptive agents is $\mathcal{U}_d = \{u_i | p_i = p_d\}$.

Let y_{il} be a random variable defined as below:

$$y_{il} = \begin{cases} 1 & \text{w.p } p_i.\theta_l + (1 - p_i).(1 - \theta_l) \\ 0 & \text{w.p } p_i.(1 - \theta_l) + (1 - p_i).\theta_l. \end{cases} \tag{1}$$

Consider the scenario when two agents u_i and u_j utilize the same service S_l and report on it. Then, based on the above notation, their reports relative to the service S_l are y_{il} and y_{jl} respectively, where:

$$\begin{aligned} Prob(y_{il} = y_{jl}) = &\ Prob[(y_{il} = 0 \wedge y_{jl} = 0) \vee (y_{il} = 1 \wedge y_{jl} = 1)] \\ = &\ Prob[(y_{il} = 0 \wedge y_{jl} = 0)] + Prob[(y_{il} = 1 \wedge y_{jl} = 1)] \\ = &\ Prob(y_{il} = 0) \cdot Prob(y_{jl} = 0) + Prob(y_{il} = 1) \cdot Prob(y_{jl} = 1). \end{aligned}$$

Throughout this paper, we shall denote $Prob(y_{il} = y_{jl})$ to be the probability that the agents u_i and u_j will agree in their appraisal. This quantity has the following property.

Theorem 1. *Let u_i and u_j two agents. If both u_i and u_j are of the same nature (either both deceptive agents or both fair), then $Prob(y_{il} = y_{jl}) > 0.5$. Similarly, if u_i and u_j are of different nature, then $Prob(y_{il} = y_{jl}) < 0.5$.*

Proof: The proof is straightforward. □

3 A LA-Based Solution to the *ATTP*

The input to our automaton, is the set of user agents $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and the reports that are communicated within the social network. With regard to the output, we intend to partition $\mathcal{U}$ into two sets, namely, the set of fair agents $\mathcal{U}_f = \{u_i | p_i = p_f\}$, and the set of deceptive agents $\mathcal{U}_d = \{u_i | p_i = p_d\}$. The intuitive principle that we use is that the agents that have the same nature (fair or deceptive) will report similar experiences about the same service, and we shall attempt to infer this phenomenon to, hopefully, migrate them to the same partition. Observe that since agents of different nature report dissimilar experiences about the same service, we hope to also infer *this*, and, hopefully, force them to converge into different partitions.

We define the Agent Migrating Partitioning Automaton ($AMPA$) as a 7-tuple as below: ($\mathcal{U}$, $\underline{\Phi}$, $\underline{\alpha}$, $\underline{\beta}$, $\mathbb{Q}$, $\mathbb{G}$, $\mathbb{L}$), where

- $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ is the set of agents.
- $\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_{2M}\}$ is the set of states.
- $\underline{\alpha} = \{\alpha_1, \alpha_2\}$ is the set of actions, each representing a group into which the elements of $\mathcal{U}$ fall.
- $\underline{\beta} = \{$'0', '1'$\}$ is the set of responses, where '0' represents a *Reward*, and '1' represents a *Penalty*.
- $\mathbb{Q}$ is the transition function, which specifies how the agents should move between the various states. This function is quite involved and will be explained in detail presently.
- The function $\mathbb{G}$ partitions the set of states for the groups. For each group, α_j, there is a set of states $\{\phi_{(j-1)M+1}, \ldots, \phi_{jM}\}$, where M is the depth of memory. Thus,

$$G(\phi_i) = \alpha_j \qquad (j-1)M + 1 \leq i \leq jM. \tag{2}$$

  This means that the agent in the automaton chooses α_1 if it is in any of the first M states, and that it chooses α_2 if it is in any of the states from ϕ_{M+1} to ϕ_{2M}. We assume that $\phi_{(j-1)M+1}$ is the most internal state of group α_j, and that ϕ_{jM} is the boundary state. These are called the states of "*Maximum Certainty*" and "*Minimum Certainty*", respectively.
- $\mathbb{W} = \{W_l^D(t)\}$, where, $W_l^D(t) = \{$ Last D records prior to instant t relative to service S_l $\}$.
  Our aim is to infer from $\mathbb{W}$ a similarity list of agents deemed to be collectively similar. From it we can, based on the window of recent events, obtain a list of pairs of the form $< u_i, u_j >$ deemed to be similar. The question of how $\mathbb{W}$ is obtained will be discussed later.

If agent u_i is in action α_j, it signifies that it is in the sub-partition whose index is j. Moreover, if the states occupied by the nodes are given, the sub-partitions can be trivially obtained using Eq. (2).

Let $\zeta_i(t)$ be the index of the state occupied by agent u_i at the t^{th} time instant. Based on $\{\zeta_i(t)\}$ and Eq. (2), let us suppose that the automaton decides a current partition of $\mathcal{U}$ into sub-partitions. Using this notation we shall later describe the transition map of the automaton. Since the intention of the learning process is to collect "similar" agents into the same sub-partition, the question of "inter-agent similarity" is rather crucial. In the spirit of Theorem 1, we shall reckon that two agents are similar if they are of the same nature, implying that the corresponding probability of them agreeing is greater than 0.5. Note that the latter parameter was assumed constant in [7].

We now consider the reward and penalty scenarios separately. Whenever two agents u_i and u_j test the same service, if their corresponding reports are identical (either both '0' or both '1'), and they currently belong to the same partition, the automaton is rewarded by moving u_i and u_j one state closer to the internal state. This mode of rewarding is called *RewardAgreeing* mode depicted in Figure 1. As opposed to this, if u_i and u_j are dissimilar and they currently belong to distinct partitions, the automaton is rewarded. This mode of rewarding is called the *RewardDisagreeingMode*.

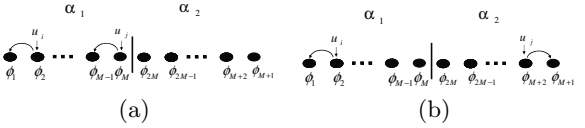

Fig. 1. (a)RewardAgreeing Mode: This is the case when both agents u_i and u_j belong to the same partition. (b) Reward Disagreeing Mode: This is the case when both agents u_i and u_j belong to different partitions.

Whenever two agents u_i and u_j test the same service, if their corresponding reports are identical (either both '0' or both '1'), and they currently belong to distinct partitions, the automaton is penalized by moving u_i and u_j one state closer to the boundary state in their respective groups. If one of the two agents is already in the boundary state, he moves to the boundary state of the alternate group. More specifically, this case is encountered when two similar agents, u_i and u_j, are allocated in distinct groups say, α_a and α_b respectively. This mode of rewarding is called the *PenalizeAgreeing* mode depicted in Figure 2.

However, if u_i and u_j are both assigned to the same subpartition, but they should rather be assigned to distinct groups, the automaton is penalized. Analogous to the above, this mode of penalizing is called the *PenalizeDisagreeing* mode, because, in this mode, agents which are actually dissimilar are assigned to the same subpartition and they are therefore penalized. This is depicted in Figure 3.

We now address the question of recording the reports associated with the various agents, which in turn, involves the set $\mathbb{W}$ defined above. In our work,

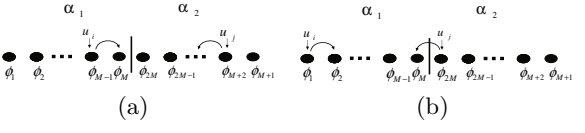

Fig. 2. PenalizeAgreeing Mode: This is the case when both agents u_i and u_j belong to different partitions. In (a), neither of them is in a boundary state. As opposed to this, in (b), the figure depicts the case when one of them, u_j, is in a boundary state.

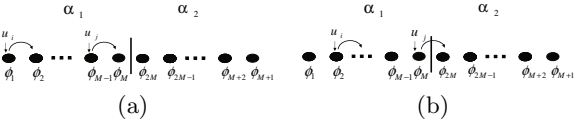

Fig. 3. PenalizingDisagreeing mode: This is the case when both agents u_i and u_j belong to the same partition, when, in actuality, they should belong to distinct partitions. In (a), neither of them is in a boundary state. As opposed to this, in (b), the figure depicts the case when one of them, u_j, is in a boundary state.

we adopt a "tuple-based" window to store the reports for a given service [8], where the window is specified in terms of the number of tuples. Observe that our approach is consistent with the work of Shapiro [9] where it was proven that in an environment in which peers can change their behavior over time, the efficiency of a reputation mechanism is maximized by giving higher weights on recent ratings and where older (stale) ratings are discounted. Clearly, this is equivalent to enforcing a sliding window model.

Despite the ability of our LA-based clustering algorithm to separate between the two groups $\mathcal{U}_f$ and $\mathcal{U}_d$, the agent can not determine which of the two groups is the fair one, $(\mathcal{U}_f)$, and which is the deceptive one, $(\mathcal{U}_d)$, unless he tries the services himself. Intuitively, if the agents knows this information, he can just take the inverse of the reports of the "liars" as being trustworthy, while he considers the rest of the reports from fair agents as also being trustworthy.

With regard to the set $\mathbb{W}$, and the decision making procedure that maximizes the likelihood of choosing high performance service, for a given service is as follows. Every agent stores the last D reports seen so far. Thus, the agent in question maintains, for every service, a sliding window over the last D reported experiences, which guarantees gathering the most recent reports. At time instant t, $W_l^D(t)$ contains the D tuples with the largest time stamps (where, if the total number d of reports seen so far is smaller than the length of the window D, the vector contains these d elements). Clearly, $W_l^D(t)$ stores the most recent D tuples[2]. Also, let $W_l^D[k]$ denote the record of index k in the vector, or the $(D-k)^{th}$ last record. Since we adopt a simple interaction model, at each time instant a random agent chooses a random service to interact with and report his experience to the rest of agents.

3.1 The Decision Making Phase

In the spirit of what we have developed so far, we assume that the services belong to two categories: High performance services and low performance services. A high performance service is a service with high value of θ, and similarly, a low performance service is a service with a low value of θ. We suppose that agent u aspires to interact with high performance services. Therefore, every time u desires to access a service, u creates a list L of the recommended services by applying a majority voting method, as explained below. Based on this list, u chooses a random service among the elements of this created list. In order to create a list of the high performance services, for every service S_l, agent u evaluates the feedback from agents that may have directly interacted with service S_l during the last D interactions. We adopt the terminology of a "witness" to denote an agent solicited for providing its feedback. In this sense, at instant t, agent u examines the service history vector $W_l^D(t)$ that contains the last reports of the witnesses regarding the performance of service S_l. For every report in the vector $W_l^D(t)$, agent u should take the reverse of the report as true if he believes that the witness

[2] In future, unless there is ambiguity, for ease of notation, we shall omit the time index t.

is a "liar", and consider the rest of the reports as being trustworthy. Following such a reasoning, given D trustworthy reports about a given service, we can apply a deterministic majority voting to determine if the service is of high performance or of low performance. Obviously, if the majority is '1', the service is assumed to be of a high performance, and consequently, it is added to the list L. However, a potential question is that of determining which partition is the deceptive one, and which involves the fair agents. In order to differentiate between the partitions we design a LA that learns which of the partitions is deceptive and which is fair – based on the result of the interaction between agent u with the selected service S_l. The automaton is rewarded whenever agent u selects a recommended service from the list L and the result of the interaction is a high performance (meaning '1'). Similarly, the automaton is penalized whenever agent u selects a recommended service from the list L and the result of the interaction is a low performance (meaning '0'). Again, we suppose that agent u in question is initially assigned to the boundary state. We observe the following:

- If agent u is in class α_j then u supposes that all the agents in α_j are fair, and the agents in the alternate class are deceptive.
- Whenever agent u decides to interact with a high performance service, he creates the list L of recommended services, and proceeds to choose a random service from L.
- If the result of the interaction is '1', a reward is generated, and the agent u goes one step towards the most internal state of class α_j.
- If the result of the interaction is '0', a penalty is generated and agent u goes one step towards the boundary state of class α_j.
- If agent u is already in the boundary state, he switches to the alternate class.

The automaton will converge to the action which yields the minimum penalty response in an expected sense. In our case, the automaton will converge to the class containing the fair agents, while the deceptive agents converge to the alternate class.

4 Experimental Results

To quantify the quality of the scheme, we measured the average performance of the selected services over all interactions, and this was used as the performance index. All the results reported have been obtained after averaging across 1,000 simulations, where every simulation consisted of 40,000 runs. The interactions between the agents and the services were generated at random, and at every time instant, a random agent was made to select a random service. In our current experimental setting, the number of agents was 20 and the number of services in the pool of available services was 100. The agent in question (i.e., the one which we are interested in) periodically accesses a service every 1,000 runs as per the above-mentioned decision making procedure.

4.1 Performance in Static Environments VS Dynamic Environments

We first report the results for environments which are static. In this particular setting, 10% of the services were high performance services with $\theta = 0.8$, and 90% were low performance services with $\theta = 0.1$. Further, 15 of the reporting agents were deceptive with $p_d = 0.2$, while 5 were fair agents with $p_f = 0.8$. The depth of memory used for the LA was $M = 10$, and the length of the sliding window was 100. Figure 4 demonstrates the ability of the approach to accurately infer correct decisions in the presence of the deceptive agents. Observe that the scheme achieves a near-*optimal* index that asymptotically approaches the performance of the high performance services, i.e., $\theta = 0.8$. To investigate the behaviour of the $AMPA$ with performances which changed with time, we first considered the scenario when these changes were made periodically. Indeed, we achieved this by changing all the service performances periodically every 5,000 runs. Further, the changes were made "drastic", i.e., by inverting them from their prior values as per $\theta_{l,new} = 1 - \theta_{l,old}$. From the results shown in Figure 5, the reader will observe that the scheme is able to adapt favourably to such changes. Indeed, from Figure 5, we notice that as the behavior of the services changed (i.e., at every $5,000^{th}$ step), the subsequent access by agent u resulted in choosing a low performance service.

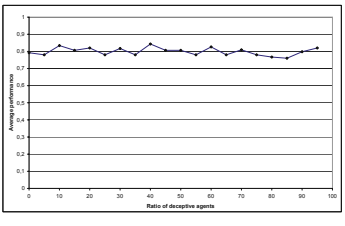

Fig. 4. The behavior of the $AMPA$, measured in terms of the average performance, in an environment when the behavior of the agents is static

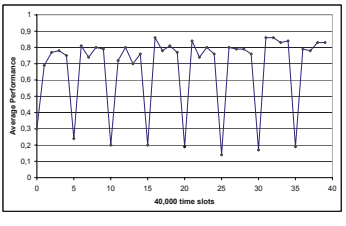

Fig. 5. The performance of the $AMPA$ with periodically changing service performances

4.2 Immunity to the Proportion of Fair Agents

We now consider the problem of investigating how "immune" our system is to the percentage of deceptive agents. Figure 6 presents the average performance of the system (over all interactions) when the ratio of deceptive agents is varied. The scheme is truly "immune" to varying the proportions of fair and deceptive agents. In fact, even if all agents are deceptive, (i.e., this is equivalent to a ratio of 100%, the average performance is stable and again achieves near optimal values that approach the index of the high performance services, $\theta = 0.8$. In our opinion, this is quite remarkable!

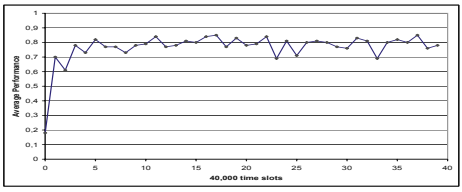

Fig. 6. The variation of the average performance under different ratios of deceptive agents

5 Conclusion

In this paper, we presented a new technique for coping with liars in reputation systems. The agents were modeled of two nature: fair and deceptive. Unlike other reported approaches, our scheme is able to counter detrimental effect of deceptive agents by intelligently combining feedback from fair and deceptive agents. The results of the simulation are conclusive and and demonstrate the potential of learning automata when applied in the area of Reputation Systems.

References

1. Dellarocas, C.: Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: Proceedings of the 2nd ACM conference on Electronic commerce, Minneapolis, Minnesota, United States, pp. 150–157. ACM, New York (2000)
2. Sen, S., Sajja, N.: Robustness of reputation-based trust: boolean case. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, part 1, Bologna, Italy, pp. 288–293. ACM, New York (2002)
3. Zacharia, G., Maes, P.: Trust management through reputation mechanisms. Applied Artificial Intelligence 14(9), 881–907 (2000)
4. Whitby, A., Jsang, A., Indulska, J.: Filtering out unfair ratings in bayesian reputation systems. In: Proceedings of the 7th Int. Workshop on Trust in Agent Societies (at AAMAS 2004). ACM, New York (2004)
5. Despotovic, Z., Aberer, K.: A probabilistic approach to predict peers performance in P2P networks. In: Cooperative Information Agents VIII, pp. 62–76 (2004)
6. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, New Jersey (1989)
7. Oommen, B.J., Ma, D.C.Y.: Stochastic automata solutions to the object partitioning problem. The Computer Journal 35, A105–A120 (1992)
8. Mayur Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows. SIAM J. Comput. 31(6), 1794–1813 (2002)
9. Shapiro, C.: Consumer information, product quality, and seller reputation. The Bell Journal of Economics 13(1), 20–35 (1982)